TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica - Campus Tecnológico Central Cartago

Escuela de Ingeniería en Computación

IC-6600 Principios de Sistemas Operativos

Prof. Armando Arce

Proyecto 2

Empacador de archivos | simple tar

Andrés David Arias Siles	2019157553
Johnny Andres Díaz Coto	2020042119

I Semestre, 2023

Lunes 29 de mayo, 2023

Tabla de Contenido

1. Introducción

El segundo proyecto del curso de sistemas operativos consiste en desarrollar un programa que simule la creación de archivos comprimidos. El objetivo principal es desarrollar un compresor de archivos que se asemeje al comando `tar` de linux, por denominarse simple-tar (`star`). Esta estrategia servirá como un acercamiento al sistema de archivos del sistema operativo, sus funcionalidades y operaciones cotidianas que suceden tras bambalinas. Se realizarán pruebas para demostrar el funcionamiento del programa. De forma general, el programa recibirá por línea de comandos el nombre del archivo comprimido con la extensión .tar y el nombre de los archivos sobre los que se desea realizar alguna ejecución..

2. Descripción del problema [1]

Se deberá programar el comando star ("simple tar", NO "estrella"), de tal forma que acepte los comandos básicos mostrados anteriormente. Para desarrollar su programa usted debe tomar en cuenta los siguientes aspectos:

- El archivo debe organizarse internamente en bloques de datos de 32K, para lo cual se deberá contar con un tabla interna tipo FAT que permita determinar la posición asignada a cada bloque del archivo. Mediante dicha tabla se podrá acceder a los datos directamente utilizando el método de acceso directo, y llevar el control de los bloques de datos.
- Al crear un archivo empacado este se crea del tamaño necesario para almacenar los archivos agregados. Cuando se borra algún contenido, el archivo empacado no cambia de tamaño sino que se lleva registro de los espacios liberados. Si posteriormente se agrega nuevo contenido entonces se reutiliza el espacio libre. Si aún así el nuevo contenido no cabe, se hace crecer el archivo empacado.
- Tome en cuenta que un archivo que se agrega puede ya existir en el archivo empacado.Es decir, lo que se desea hacer es actualizar su contenido. Para esto existe la opción update (-u) que sobrescribirá el contenido de un archivo.
- Se debe llevar un control de los espacios libres, para asignar los huecos libres se utilizará la técnica del primer ajuste o el siguiente ajuste.
- Este programa no utilizará los derechos de acceso, que normalmente almacenaría un archivo empacado tar en ambiente UNIX.

- La opción de desfragmentación (-p) no es estándar (no está presente en tar) y lo que hace es desfragmentar el contenido almacenado en el archivo empacado y liberar cualquier espacio sin utilizar. Es decir, con este comando se liberarán todos los bloques libres y el tamaño del archivo empacado se ajustará al contenido real existente.
- La opción -v muestra información sobre la operación ejecutada. Se puede aplicar dos veces -vv para ver información adicional.

```
tar -rvf foo.tar test.doc
```

3. Detalle de la Solución

La solución se divide en cuatro funcionalidades principales, explicadas en la sección 3.2. El programa empaca o comprime (como se le menciona en adelante dentro de esta documentación) en un archivo binario con extensión .tar un conjunto de archivos que son especificados por el usuario, y que se encuentren en el mismo directorio que el ejecutable. Para ello, se crean estructuras que permiten el almacenamiento temporal de los datos mientras se lleva a cabo el proceso de compresión y extracción de los archivos.

Al crear un archivo comprimido, se envía el nombre deseado y los archivos por incluir a la función comprimir, que se encargará de escribir la información de estos archivos en uno solo. En esta función, se lee cada archivo, que serán guardados en bloques de datos de 32 KB cada uno. Durante este proceso, se lleva un registro de los bloques de datos asociados a un archivo particular, mediante una tabla directorio y una tabla FAT, cuyas estructuras se detallarán más adelante. Conforme se van consumiendo los archivos por comprimir, se escriben en el tar, y cuando ya todos hayan sido procesados, se escriben las tablas directorio y fat, al final del archivo binario tar.

Al descomprimir, se lee primeramente estas tablas del directorio y FAT, para que así se pueda llevar el orden adecuado en que deben extraerse los archivos.

Para hacer uso del programa, dentro de la línea de comandos, hay que ubicarse en el directorio que corresponde (donde está el ejecutable), y ejecutar el comando de la siguiente forma:

```
tar <opciones> <archivoSalida> <archivo1> <archivo2> ... <archivoN>
```

3.1. Estructuras de Datos

A continuación, se presenta una descripción general de las principales estructuras de datos utilizadas en la solución:

- **DIRECTORY**:

  Esta estructura cumple la función de ser el directorio de archivo para la tabla FAT, por lo que dentro de ella se encuentran los datos primitivos de un archivo, que son:

  - File: Arreglo de caracteres que contienen el nombre del archivo.
  - Primer bloque de información, dentro de la tabla FAT.
  - Size: Cuantos bloques de información conforman el archivo.

- **FAT:**

  De las siglas de File Allocation Table, es una estructura de datos que guarda en qué bloque de información del disco se guarda la información del archivo, esta estructura está compuesta por:

  - id: Posición dentro del FAT.
  - next: Cual es el bloque de información siguiente.

- **BLOCK:**

  Simulación del bloque de información de un disco, por lo que se conforma por dos campos:

  - blockId: Número del bloque de información.
  - data: Información que guarda este bloque. Para este caso se usa un tamaño de 32 Kb (32768 bytes)

Al final estas estructuras se usan en combinación para poder recrear un sistema de archivos que controla un archivo tar. Su uso aplicado de la manera más general sería, buscar el archivo en la estructura directory para así saber en qué bloque inicia la informacion y cuantos bloques me toma llegar hasta el final, para una vez obtenido el primer bloque, se revisa dentro de la tabla FAT para buscar en qué bloque del disco se encuentra la información y finalmente se lee la información del disco.

3.2. Componentes del programa

3.2.1. Mecanismo de acceso a archivos

Los archivos por comprimir, deben estar en el mismo directorio donde se encuentre el ejecutable. Propiamente para acceder a los archivos, se realizan operaciones sobre archivos de forma binaria, lo que hace posible trabajar con el contenido puro de los archivos, para que puedan ser comprimidos o extraídos. Se utilizan las siguientes funciones:

- fopen: Abre el archivo binario, en modo de escritura, lectura o ambos
- fread: Permite obtener información del archivo, la cual se almacena en un buffer de tamaño predeterminado.
- fwrite: Realiza una lectura del archivo, de determinada cantidad de bytes, los cuales son alojados en un buffer
- fseek: Posiciona el cursor de lectura/escritura en una posición específica
- fclose: Cierra el archivo de trabajo.

3.2.2. Tabla Interna:

Para una correcta aplicación de los principios buscados en el proyecto, se utilizan 3 tablas internas, de forma tal que sea posible realizar una Asignación Enlazada.

- directoryTable: Representa el sistema de archivos, utiliza la estructura DIRECTORY, le forma tal que almacena el nombre de los archivos contenidos, junto a las referencias básicas para estructurar el archivo
- fatTable: Representa la tabla de asignación de archivos, la cual tiene una entrada por cada bloque de disco, y se encuentra debidamente indexada; funciona como una lista enlazada de nodos FAT. Debido a la naturaleza del proyecto, es posible cargarla en su totalidad a memoria. Es importante señalar que se utiliza un valor de ‘0’ en el siguiente bloque, para expresar que es el último bloque que compone el archivo.

- dataBlocks: Contendrá todo el contenido de los archivos comprimidos, distribuido a través de los bloques (BLOCK de 32 Kb). Para poder almacenar/recuperar correctamente los archivos, se debe recorrer esta tabla, según la indexación indicada por el fatTable

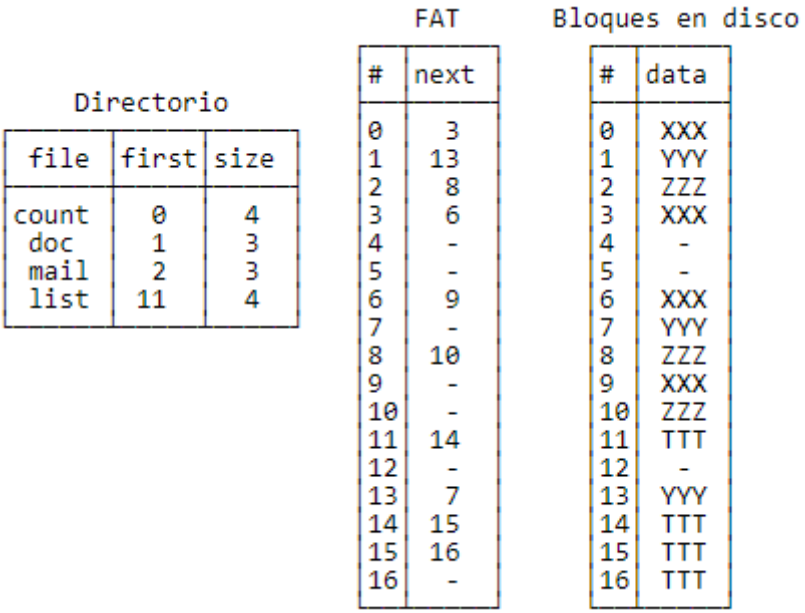


Imagen 1. Diagrama que representa el sistema de archivos utilizado en el proyecto. (Fuente: Página del curso) [2]

3.2.3. Estrategia de administración de espacios libres:

La estrategia para asumir el reto es usar códigos propios del programa, algo que puede diferir de las demás implementaciones, puesto que estos códigos fueron impuestos por el equipo de trabajo.

En ese caso, las estructuras de datos se inicializan con valores nulos en caso de los nombres de archivos, mientras que los campos que contienen números enteros se inicializan con un valor de -1, en este caso este valor indica cuando un espacio no se encuentra en uso y se encuentra libre.

Dicho esto, al iniciar el programa, como se menciona, las estructuras se inicializan con valores nulos y -1, por lo que para asignar un espacio se busca por estos valores para saber qué espacios se encuentran libres y se asigna la memoria o el registro al archivo que lo solicita en su momento.

En el caso de asignación de estos espacios se logra gracias a la función *getBloqueDisponible* y *getSigBloqueDisponible*, donde retorna como valor un bloque libre

siguiendo las técnicas de primer ajuste y siguiente ajuste, por lo que el valor retornado puede ser usado para asignación.

En el caso de borrar algún archivo contenido en el tar, simplemente se vuelven a inicializar los valores del archivo en cuestión a valores nulos y -1, para que así se puedan identificar como espacios disponibles nuevamente.

3.2.4. Procedimiento de desfragmentación del archivo:

Objetivos, parámetros, resumen de cómo funciona y los resultados esperados, importancia.

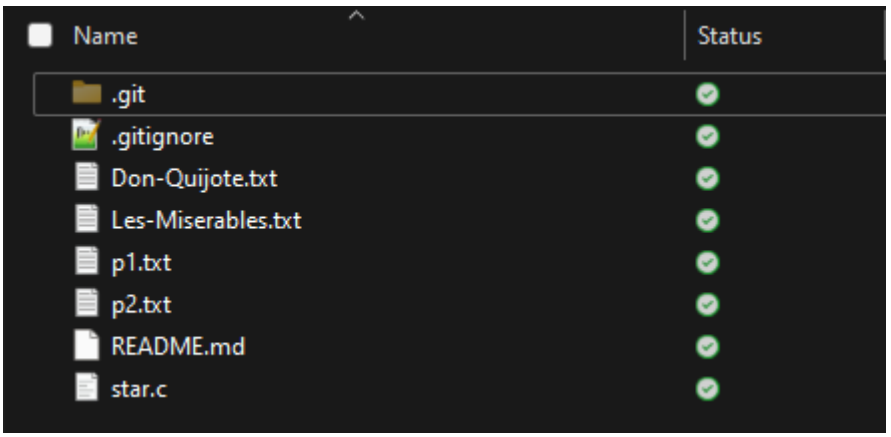
4. Pruebas

Durante el desarrollo del software, se realizaron esencialmente 4 tipos de pruebas. En cada caso, se utilizaron los archivos de prueba descritos en el anexo I.

1) Empacar archivos: Se comprimen los 3 archivos en uno solo, y se verifica que en efecto el archivo resultante contenga la información de los 3 archivos.

```
./star -cvf textos.tar p1.txt p2.txt Don-Quijote.txt Les-Miserables.txt
```

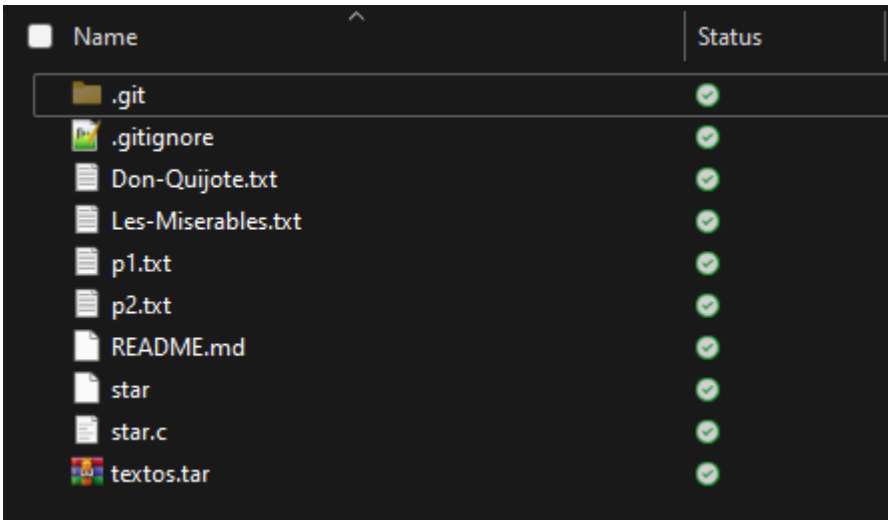
Observe el estado **antes** de ejecutar el comando anterior:



Observe el **resultado** de la ejecución:

```
Comprimir 4 archivos

 -> Iniciando compresion
    procesando p1.txt a partir del bloque 0
    procesando p2.txt a partir del bloque 1
    procesando Don-Quijote.txt a partir del bloque 2
    procesando Les-Miserables.txt a partir del bloque 70
 -> textos.tar creado!
---------
Habilitada la opción de reporte detallado
```



2) Listar Archivos: Se ejecuta el comando, esperando como respuesta una impresión de la tabla de directorio

```
./star -tvf textos.tar
```

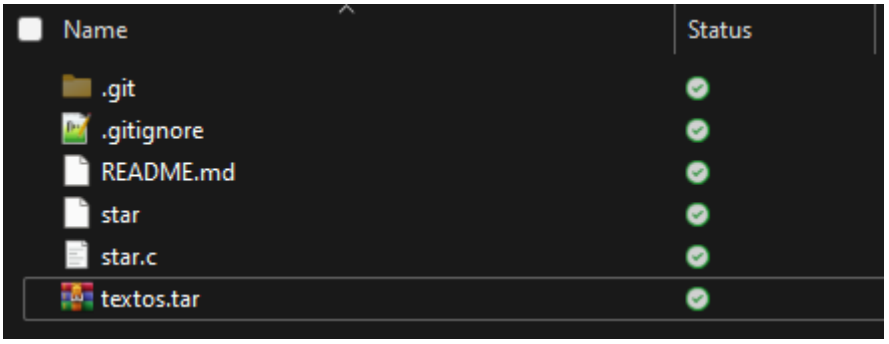
Observe los resultados de la prueba:

```
/mnt/c/U/j/OneDrive /T/I/S/C/Proy/IC6600_SOperativos_Proyecto  proyecto_02 !1 ?7  ./star -tvf textos.tar
{ p1.txt | 1 bloques }
{ p2.txt | 1 bloques }
{ Don-Quijote.txt | 68 bloques }
{ Les-Miserables.txt | 103 bloques }
Habilitada la opción de reporte detallado
```

3) Desempacar archivos: El archivo creado anteriormente, se procede a descomprimir. Este escribe correctamente los archivos que estaban contenidos, en el directorio actual

```
./star -xvf textos.tar
```

Primero se eliminan los archivos para no interferir con el resultado:



Luego se procede con la prueba y se obtiene:

```
/mnt/c/U/j/OneDrive /T/I/S/C/Proy/IC6600_SOperativos_Proyecto  proyecto_02 !1 ?7  ./star -xvf textos.tar

 -> Descomprimiendo textos.tar!

(p1.txt | 0 | 1 ) { (0 | 0) }

(p2.txt | 1 | 1 ) { (1 | 0) }

(Don-Quijote.txt | 2 | 68 ) { (2 | 3) -->(3 | 4) -->(4 | 5) -->(5 | 6) -->(6 | 7) -->(7 | 8) -->(8 | 9) -->(9 | 10) -->(1
9) -->(19 | 20) -->(20 | 21) -->(21 | 22) -->(22 | 23) -->(23 | 24) -->(24 | 25) -->(25 | 26) -->(26 | 27) -->(27 | 28) -
 | 37) -->(37 | 38) -->(38 | 39) -->(39 | 40) -->(40 | 41) -->(41 | 42) -->(42 | 43) -->(43 | 44) -->(44 | 45) -->(45 | 4
>(54 | 55) -->(55 | 56) -->(56 | 57) -->(57 | 58) -->(58 | 59) -->(59 | 60) -->(60 | 61) -->(61 | 62) -->(62 | 63) -->(63

(Les-Miserables.txt | 70 | 103 ) { (70 | 71) -->(71 | 72) -->(72 | 73) -->(73 | 74) -->(74 | 75) -->(75 | 76) -->(76 | 77
(85 | 86) -->(86 | 87) -->(87 | 88) -->(88 | 89) -->(89 | 90) -->(90 | 91) -->(91 | 92) -->(92 | 93) -->(93 | 94) -->(94
 | 103) -->(103 | 104) -->(104 | 105) -->(105 | 106) -->(106 | 107) -->(107 | 108) -->(108 | 109) -->(109 | 110) -->(110
8) -->(118 | 119) -->(119 | 120) -->(120 | 121) -->(121 | 122) -->(122 | 123) -->(123 | 124) -->(124 | 125) -->(125 | 126
>(133 | 134) -->(134 | 135) -->(135 | 136) -->(136 | 137) -->(137 | 138) -->(138 | 139) -->(139 | 140) -->(140 | 141) -->
 | 149) -->(149 | 150) -->(150 | 151) -->(151 | 152) -->(152 | 153) -->(153 | 154) -->(154 | 155) -->(155 | 156) -->(156
4) -->(164 | 165) -->(165 | 166) -->(166 | 167) -->(167 | 168) -->(168 | 169) -->(169 | 170) -->(170 | 171) -->(171 | 172

 -> Se ha extraido el archivo!
---------
Habilitada la opción de reporte detallado
```

Al ser con el reporte detallado se muestra cuales son los bloques donde se encuentra la información del archivo dentro del comprimido.

4) Eliminar un archivo del tar: Se realiza la eliminación del

```
./star -rvf textos.tar Don-Quijote.txt
```

En esta prueba se muestra la organizacion de la tabla fat, por lo que se muestra la parte de las seccion correspondiente a Don-Quijote.txt que ha sido el archivo eliminado:

```
/mnt/c/U/j/OneDrive /T/I/S/C/Proy/IC6600_SOperativos_Proyecto proyecto_02 !1 ?7 ./star -dvf textos.tar Don-Quijote.txt

 -> eliminando Don-Quijote.txt!
(0 | 0)
(1 | 0)
(2 | 3)
(3 | 4)
(4 | 5)
(5 | 6)
(6 | 7)
(7 | 8)
(8 | 9)
(9 | 10)
(10 | 11)
(11 | 12)
(12 | 13)
(13 | 14)
(14 | 15)
(15 | 16)
```

```
FAT table despues de eliminar:

(0 | 0)
(1 | 0)
(2 | -1)
(3 | -1)
(4 | -1)
(5 | -1)
(6 | -1)
(7 | -1)
(8 | -1)
(9 | -1)
(10 | -1)
(11 | -1)
(12 | -1)
(13 | -1)
(14 | -1)
(15 | -1)
(16 | -1)
(17 | -1)
(18 | -1)
(19 | -1)
(20 | -1)
(21 | -1)
(22 | -1)
(23 | -1)
(24 | -1)
(25 | -1)
```

Note que a partir del bloque 2 se tienen valores de -1, lo correspondiente a lo que se dijo en la sección anterior, donde un -1 indica que es un espacio libre.

Para comprobar de manera definitiva se lista el contenido del comprimido, teniendo como resultado:

```
/mnt/c/U/j/OneDrive /T/I/S/C/Proy/IC6600_SOperativos_Proyecto proyecto_02 !1 ?7 ./star -tvf textos.tar
{ p1.txt | 1 bloques }
{ p2.txt | 1 bloques }
{ Les-Miserables.txt | 103 bloques }
```

Note como Don-Quijote.txt ya no se encuentra en la lista.

5. Conclusiones

Después de realizar las pruebas correspondientes, se puede concluir que el sistema de asignación enlazado de archivos es factible, y brinda resultados certeros. También se ha observado que es necesario contar con algún estimado del tamaño y la cantidad de los archivos que se pretenden empacar, esto debido a que esta información es necesaria para definir las estructuras, esto al menos en lenguajes de programación como lo es C. El star desarrollado tiene capacidad de 256 bloques de 32 kb cada uno, dando una capacidad máxima de 8 mb.

A pesar de que la lógica detrás de la teoría sea bastante intuitiva, a la hora de llevarla a la programación, requiere de especial cuidado para no perder de vista detalles que podrían causar problemas al mecanismo para empacar y desempacar archivos. La teoría a pesar de no ser reciente, aún es situable en los sistemas modernos.

Se sugiere explorar distintos tamaños de bloque (ej. 16 Kb, 64 Kb, etc) con el fin de evaluar otras alternativas, así como otras metodologías para el manejo de la tabla fat; o inclusive, otros métodos de asignación, con el fin de comparar resultados.

6. Referencias

[1] A. Arce, “Segundo Proyecto”, IC6600 Principios de Sistemas Operativos - TEC, https://operativos-tec.netlify.app/proyectos/Proyecto2.html (Consultado 18 may., 2023)

[2] A. Arce, “Interfaz del sistema de archivos”, IC6600 Principios de Sistemas Operativos - TEC, https://operativos-tec.netlify.app/temas/Cap10_interfaz_archivos.html (Consultado 23 may., 2023)

[3] A. Arce, “Sistema de Archivos”, IC6600 Principios de Sistemas Operativos - TEC, https://operativos-tec.netlify.app/temas/Cap11_sistemas_archivos.html (Consultado 23 may., 2023)

[4] IBM Documentation, “fseek() – fseeko() — Reposition File Position”, https://www.ibm.com/docs/en/i/7.4?topic=functions-fseek-fseeko-reposition-file-position (Consultado 18 abril, 2023)

7. Anexos

7.1. Anexo 1: Estructura del Proyecto de Entrega

El entregable enviado al aula virtual del curso de IC-6600 contiene lo siguiente:

- **Documentación**: Este archivo en formato pdf para una fácil lectura del mismo.

- **star.c**: Código fuente del programa desarrollado por el equipo de trabajo que cumple con lo solicitado para el proyecto en cuestión.

- **Archivos de Pruebas**: Esta carpeta contiene los archivos que fueron utilizados para las pruebas durante y después del desarrollo del software.

    - **p1.txt**: Pequeño archivo de prueba cuyo tamaño no supera 1 KB.
    - **p2.txt**: Pequeño archivo de prueba cuyo tamaño no supera 1 KB.
    - **Don-Quijote.txt**: Archivo txt de la obra de Miguel de Cervantes, Don Quijote de la Mancha. Obtenido mediante el proyecto Gutenberg
    - **Les-Miserables.txt**: Archivo txt de la obra. Obtenido mediante el proyecto Gutenberg

      https://www.gutenberg.org/

7.2. Anexo 2: Mini manual de usuario

Consideraciones:

- Los nombres de los archivos (tanto el comprimido como los empaquetados) no debe superar los **256 bytes***
- La capacidad máxima de almacenaje del tar, es de **8 192 KB**. Archivos que superen esta cifra, quedarían incompletos*
- Los archivos por empaquetar deben estar ubicados en el mismo directorio que el ejecutable, además, al extraer archivos estos quedarán en el directorio actual.

- La indicación de los flags/opciones del programa por línea del comando, por el momento son solo soportadas mediante la versión de una sola letra. (Por ejemplo: -cvf es soportado, mientras que --create --verbose --file, no). Adelante se detalla ejemplos de cómo utilizar el programa desde la línea comando

Uso del programa:

Crear un archivo empacado

```
./star -cvf <filename.tar> <file1> <file2> ... <file n>
```

Listar Archivos:

```
./star -tvf <filename.tar>
```

Desempacar archivos:

```
./star -xvf <filename.tar>
```

Eliminar un archivo del tar: Se realiza la eliminación del

```
./star -rvf <filename.tar> <filename>
```

* Para solucionar ambas condiciones, bastaría cambiar las variables globales definidas dentro del código, a un tamaño que satisfaga las necesidades.

```
//  Variables globales
// --------------------
#define BLOCK_SIZE     32768
#define FAT_SIZE       256
#define DIRECTORY_SIZE 256
#define MAX_DATA_BLOCK 256
#define MAX_NAME_LEN   256
```